情報メディア 29－3
（1997. 1. 10）

拡張トランザクションモデルに基づくサービス連携方式

藤村 考　　寺田雅之
NTT 情報通信研究所
{fujimura,terada}@isl.ntt.co.jp

概要 本論文では、広域分散環境において提供されている自立したサービスを連携させ処理するための、*TxBridge* と呼ぶサービス連携システムを提案する。*TxBridge* では、一つの連携処理を各サービスに対して要求するトランザクションの集合と、それらのトランザクション間の制約条件として表現し、制約充足エンジンによるトランザクションのスケジューリングを行う。また、*TxBridge* は IntelligentPad 上に構築された drag&drop に基づく GUI を利用することにより、トランザクション間でのデータの受渡しと制約条件を簡易に定義すること可能にする。

Service Coordination System Based on Advanced Transaction Model

Ko Fujimura　　Masayuki Terada
NTT Information and Communication Systems Labs

Abstract This paper proposes a services coordination system, called *TxBridge*, that enables complex processing composed of a set of tasks executed for autonomous services in an open distributed environment. We define composite processing as a set of transactions for the services and the constraints among the transactions. The *TxBridge* system schedules the execution sequence of the transactions using constraints-solving techniques. The *TxBridge* system also enables users to easily define the passing of data and constraints among the transactions using drag & drop-based GUI, built on top of the IntelligentPad system.

1．はじめに

近年、World Wide Web(WWW)の急速な普及により、様々なサービスがインターネット等の広域分散ネットワーク（以後、単にネットワークと呼ぶ）上に出現している。これにより例えば、ホテルや航空券などの各種予約や、商品の購入等がネットワークを介してできるようになってきた。このようなサービスの増加につれ、いくつかのサービスを連携させて、より複雑な処理を行いたいという要求が高まっている。典型例としては、ホテルや航空券の予約を組合せて、旅行に必要な宿泊施設と交通手段を予約する場合等がある。

連携する対象のサービスがこのような旅行の予約に限定できる場合には、現実社会には旅行代理店が存在し、それらの予約を旅行代理店に依頼することができる。しかし今後、多様なサービスがネットワーク上で出現されるようになるに従い、旅行に限らない汎用的なサービスの連携が求められるようになる。例えば、ある会議に出席するため航空券を予約したいが、とれない場合にはテレビ会議室を予約するという場合等である。この場合、航空券の予約がとれない場合はキャンセル待ちを行うと共に、テレビ会議室の予約を押えておき、キャンセル待ちに成功した場合には、テレビ会議室をキャンセルするといった連携処理が必要となる。

このような要求に応えるため、筆者らは、*TxBridge* と呼ぶネットワーク上のサービスを提供することを目指している。*TxBridge* は、ネットワーク上の複数のサービスを連携させた処理の要求

を利用者から受け付け、利用者に代わってその連携処理の実行を行うサービスである。しかし、このような連携サービスを実現するには、次のような問題がある。

（1）複数のサービスに対する連携処理の中には、例えば、商品の発送処理と代金の課金処理のように、両方の処理が必ず行われることを保証（原子性の保証）することが要求される場合がある。従来、このような一貫性を保証するためには、2相コミットプロトコル（2PC）[1]等のトランザクション処理技術が使われてきたが、本研究が前提とするネットワーク上では、通信経路の輻輳等による通信の遅延や、各サービスの処理時間が予想できない等の理由により、処理の開始から終了までの期間が長期化する場合が多く、この場合は2PCでは資源が長時間占有され効率が低下するという問題がある。また、2PCを利用するには、連携される対象のすべてのサービスが2PCの制御に必要なインタフェースを公開しなければならないが、*TxBridge*では、任意のサービスの連携を目標とするため、このような前提は置けない。

（2）複数のサービス間で連携処理を行うためには、サービス間で各種データの受渡しができなくてはならない。しかし、各サービスは独立に設計、運用される自立した主体であり、そのサービスに対する入出力データの型は統一されておらず、任意のサービス間でデータの受け渡しを行うのは困難である。

（3）あるサービスの処理に失敗した場合、代替処理を別のサービスに依頼する等の連携処理のシナリオは、複雑な条件を伴う場合が少なくない。従来の研究では、このような条件は論理式やスクリプト言語を用いて記述していた。しかし、連携処理のシナリオは末端の利用者がその処理を依頼する都度与えられるものであるため、論理式やスクリプト言語では難しすぎる。

本論文では、上記（1）の問題を解決するため、まず、連携処理全体を一つのトランザクションとすることを前提とせず、連携対象の各サービスに対する単位処理を各々トランザクションとしてモデル化する。そしてこの場合、一つの連携処理は、複数のトランザクションから構成されることになるので、これらの複数のトランザクション間の制約条件を*TxBridge*に与え、*TxBridge*がその関係を満たすようにトランザクションの実行をスケジュールすることで原子性を保証する。

また、上記（2）の問題を解決するため、まず、各サービスが提供する多様なインタフェースの違いを吸収し、統一的なインタフェースを提供するソフトウェア部品を作り、それをWWWにより利用者に配布することにした。そして、利用者はこれらの部品を組合せることにより、任意のサービス間でのデータの受渡しを定義する。また、このアプローチにより連携対象のサービスが提供する既存のインタフェースには手を入れなくて済む。

また、上記（3）の問題を解決するため、上記で述べた統一的なインタフェースを提供するソフトウェア部品をdrag&drop操作に基づき貼り合わせることで、データの受渡し、トランザクション間の依存関係等の連携処理のシナリオを簡便に定義する方式を提案する。

*TxBridge*は、現時点ではまだ開発中である。そこで本論文では、*TxBridge*のアーキテクチャについて述べ、実現方式の詳細はおよびその評価については別途報告する。

以下、2章では、*TxBridge*の基本となる拡張トランザクションモデルを述べ、従来の研究との違いを明らかにする。3章では、*TxBridge*のアーキテクチャについて述べる。4章では、*TxBridge*を利用するためのGUIについて述べる。5章はまとめである。

2．拡張トランザクションモデル

前章で述べたように、*TxBridge*のような汎用的なサービス連携を実現するための第一の問題点は、任意のサービス間で共通に利用可能な一貫性保証プロトコルの存在を仮定できないことである。このような問題は、マルチデータベースシステム(MDBS)や拡張トランザクションモデルの分野で、近年、盛んに議論されている[1]。トランザクションは、一般に、原子性、一貫性、分離性、永続性

のACID属性[1]をもつ処理として定義されるが、上記の問題により、連携処理全体を一つのACID属性をもつトランザクションとすることは現実的でない。そこで、ASSET[2]、Flex[3]、ConTract[4]等の拡張トランザクションモデルでは、一つの連携処理を複数のトランザクションの連携処理とみなし、それらのトランザクション間の関係を保証する方法を提案している。これらの中で、ASSET、ConTractは、トランザクション間の関係を処理のフローに基づき定義するのに対し、Flexは、トランザクション間の依存関係を論理式に基づき宣言的に定義し、実際の処理のフローはFlexシステムが自動的に導出するという特徴がある。連携の条件が複雑になると、最適な処理のフローを利用者が与えるのが困難になるため、Flexのように処理のフローを明示的に与えない方式が望ましい。したがって、*TxBridge*ではFlexを方式を適用することにした。

Flexトランザクションは、正確には、次の5つの組 $T(B, S, F, \Pi, f)$ として与えられる。ただし、

- B はトランザクションの集合$\{t_1,\ldots,t_n\}$である。
- S はトランザクション間の成功に関する半順序制約である。$t_i \rightarrow t_j \in S$であるとき、$t_j$ が実行されるためには、t_iの成功が条件となる。
- F はトランザクションの失敗に関する半順序制約、$t_i \rightarrow t_j \in F$であるとき、$t_j$ が実行されるためには、t_iの失敗が条件となる。
- Π はトランザクションの外部依存の実行順序制約である。例えば、t_iの実行が日付D以降であることは、$D \rightarrow date(t_i)$等で表わされる。
- fはTの受理関数と呼ばれ、FlexトランザクションTを実行した結果、Tが成功するかどうかを規定するものである。例えば、$f(x_1,x_2,x_3,x_4)=(x_1 \wedge x_3) \vee (x_2 \wedge x_4)$と定義されたとき、$T$は、$t_1$および$t_3$、あるいは$t_2$および$t_4$が成功したときに受理される。ここで、$x_1,\ldots,x_n$は$t_1,\ldots,t_n$ $(t_i \in B)$の実行状態である。

しかし、Flexはモデルとしては利用できるが、多様なサービス間でデータの受け渡しをどのように実現するかという第1章で示した問題（2）を解決するものではない。また、Flexで提案されている制約定義言語は述語論理をベースとする専用の言語であり、これは末端の利用者が記述するものとしては難しすぎ、第1章で示した問題（3）を解決するものでもない。

一方、（2）の問題を解決する技術としては、Active-X[5]、IntelligentPad[6]等のコンポーネントウェアがある。コンポーネントウェアはソフトウェア部品の組み立てによりソフトウェアあるいは複合メディアを開発する体系であり、各部品がサポートしなくてはならないインタフェースを詳細に規定することにより、予め独立に開発された部品間でのデータの連携を可能にしている。しかし、これらは元々ソフトウェア開発のために設計されたものであるため、部品の組合せには各部品のインタフェース等の知識を前提とし、*TxBridge*のように、末端の利用者が処理を依頼する都度与える「使い捨て」の連携処理の定義に利用するには、やはり複雑すぎる。そこで、*TxBridge*では、これらのコンポーネントウェアのデータ連携方式を利用し、これらの上に、末端の利用者が連携処理のシナリオを容易に定義できるGUIを新たに開発することにした。具体的には、Flexに基づくトランザクションの集合とトランザクション間の制約条件を生成を、drag&drop操作のみで容易に定義できるようにする。

3. *TxBridge*のアーキテクチャ

3.1 構成要素

本章では*TxBridge*のアーキテクチャを提案する。本サービスシステムの構成要素としては、連携対象のサービス、利用者、およびこれらを仲介者から構成される。仲介者はさらにコーディネータ、コーディネータ・ファクトリ、インタフェース・ブローカの3種類の要素から構成する。これらの要素は以下のように定義する。

サービス

ネットワーク上のプロセスであり、公開されたトランザクションを介してのみプロセスの内部状態を参照、更新することができる。

利用者

複数のサービスに対するトランザクションの集

合と、それらのトランザクション間の制約条件の組から構成される連携処理定義を生成するネットワーク上のプロセス。

コーディネータ

利用者から与えられた連携処理定義に基づき、各サービスに対するトランザクションの実行のスケジューリングを行い、トランザクションの実行状態を監視し、最終的に与えられたトランザクション間の制約条件を満たすように、トランザクションの実行を制御するネットワーク上のプロセス。

コーディネータ・ファクトリ

上記、コーディネータを利用者から与えられた一つあるいは複数の連携処理毎に動的に生成し、ネットワーク上に配置するネットワーク上のプロセス。

インタフェース・ブローカ

トランザクション定義フォーム(TDF)を利用者に配布するネットワーク上のプロセス。ここで、トランザクション定義フォームとは、各々のサービスのトランザクション毎に存在し、このフォームにデータを入力することにより、利用者がトランザクション間の連携を定義するためのソフトウェア部品（図１の網掛部）を生成するものである。これらの詳細と使い方は４章で述べる。

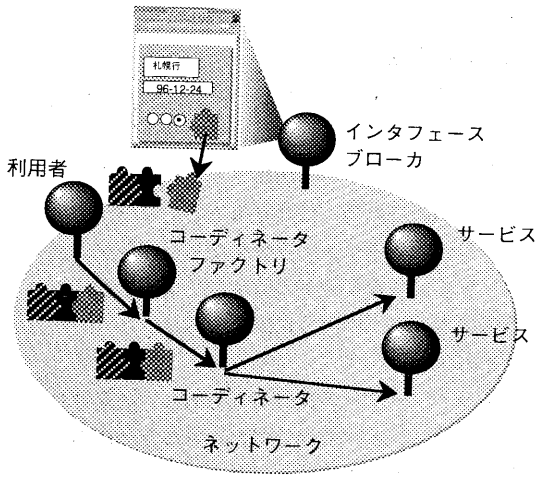

図１構成要素

3.2 処理の流れ

Step1: 利用者による連携処理定義

利用者はインタフェース・ブローカから、トランザクション定義フォームをダウンロードし、４章で述べるGUIに基づき、以下の３つの要素から成る連携処理定義を作成する。

(1) 連携処理を構成する各トランザクション(Flexにおけるトランザクション集合B)

(2) トランザクション間のデータの連携（Flexでは明示的にモデル化していないが、Bに含む）

(3) トランザクション間の制約条件C　(Flexにおける制約条件S, F, Π, f)

Step2: コーディネータの生成

利用者は、コーディネータ・ファクトリにStep1で得られた連携処理定義を送付する。コーディネータ・ファクトリは、コーディネータのインスタンスを生成し、そのオブジェクトに連携処理定義を受け渡す。本論文では、連携処理毎に一つのコーディネータが生成されるものとする。

Step3: 候補集合の決定とスケジューリング

連携処理定義を受け取ったコーディネータは、$B=\{t_1,\ldots,t_n\}$の部分集合で、トランザクション間の制約条件Cを充足できるあるトランザクション候補集合CB_iとその実行順序を決定し、各サービスに対しトランザクションの処理依頼を行う。

CB_iとその実行順序を効率的に決定する方法については、本論文では述べないが、これは制約充足問題[7]として扱うことができ、それらの成果を適用することが可能である。ただし、トランザクションの取り消し可能期限等の制約は、サービス依存であるため、これらの違いを考慮してスケジュール行うことが必要となる。これらの制約は４章で述べるTPadの中にカプセル化されるため、コーディネータにも渡される。コーディネータは、これらを制約条件に加えスケジューリングを行う。

Step4: 各トランザクションの実行

コーディネータからトランザクションの処理要求を受け取った各サービスは、その処理を実行し、結果をコーディネータに返す。ただし、この段階では各$t_x \in CB_i$は、コーディネータより最終的なコ

ミット指示があるまで、取り消し可能な状態にしておく。具体的には、そのトランザクションがデータ資源の更新を行なう場合には、仮更新、すなわち、資源のオリジナルをロックし、コピーを更新する方法や、コーディネータからのロールバック指示を受け取った場合に、トランザクションの効果を消す補償トランザクション[1]を実行させる機構を提供するという方法がある。

Step5: 制約条件充足チェック

コーディネータは、一つ以上のトランザクションの実行の失敗が返されたとき、あるいは処理を依頼した全てのすべてのトランザクションの実行結果が揃ったとき、制約条件 C を満たすかどうかをチェックする。制約条件を充足できれば、Step7に、充足できなければ、次のステップに進む。

Step6: 候補集合の再決定と再スケジューリング

もし、一つでも制約条件を満たすことができなければ、コーディネータは、新しい候補集合 CB_{i+1} とその実行順序を定める。もし、過去に確認していない新しい候補集合が見つからないときはその連携処理全体が失敗となる。

そして、コーディネータは、$\{t_x \mid t_x \notin CB_{i+1}$ かつ $t_x \in CB_i\}$ の要素をロールバックの指示をし、$\{t_x \mid t_x \in CB_{i+1}$ かつ $t_x \notin CB_i\}$ の要素の実行を各サービスに要求する。そして、Step4に戻り、新たなトランザクションの実行を行なう。

Step7: トランザクションのコミット

コーディネータは、各サービスで処理され、成功した全ての $t_x \in CB_i$ に対し、コミットを指示を行う。各サービスは、コーディネータからのコミット指示に従い、仮更新されている資源を実更新すること等、コミット処理を行う。

Step8: コーディネータの消滅

コーディネータは、各サービスのコミット処理が正常に終了するのを確認し、自己を消滅させる。

4. GUI

3.2 で述べたように、利用者は、(1)連携処理を構成する各要素トランザクション、(2)トランザクション間のデータの連携、(3)トランザクション間の制約条件、を定義しなければならない。

このため、*TxBridge* では、以下の４種類の pad[6] として可視化されたソフトウェア部品を用いる。

Transaction definition form (TDF) 連携処理を構成する各要素トランザクションを定義するフォームである。そのフォームにトランザクションの入力データを記入することにより、以下に示すTPadとPPadと呼ぶ2種類のオブジェクトを作り、TDF の上に表示する（図２）。尚、TDF は、インタフェース・ブローカにより、WWW 等により利用者に配布することを前提にしている。

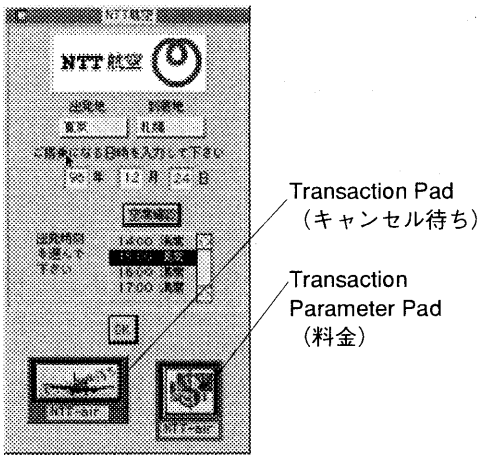

図2 トランザクション定義フォーム

Transaciton pad (TPad) トランザクションをパッドとして視覚化したオブジェクトである。このオブジェクトのインタフェースとしては、トランザクションを実行するサービスのリファレンス *SID*、実行するトランザクションの識別子 *TID*、トランザクションの内容を定義するデータのリスト $P=<p_1,\ldots,p_n>$、トランザクションの取り消し可能期限等の実行制御情報 C の４つ組$<SID,TID,P,C>$を持つ。TPad を以下で述べる CPad のトランザクション貼付域に drag&drop することにより、トランザクション間の関係を定義することができる。

Transaction parameter pad (PPad) トランザクションの実行結果得られる（であろう）出力データを視覚化したオブジェクトである。どのデー

タをPPadとして利用者に提供するかということは、TDFの設計者に委ねられている。このオブジェクトのインタフェースとしては、トランザクションを実行するサービスのリファレンス*SID*、トランザクションの識別子*TID*、トランザクションの出力データp、出力データの型T_p、4つ組<*SID,TID,p,*T_p>を持つ。PPadをデータを受渡す対象の他のトランザクションを定義するためのTDFのデータ入力部にdrag&dropすることにより、データの連携を定義することができる。

Constraint definition pad (CPad) トランザクション間の関係を定義するためのオブジェクトである。TPad貼付域とトランザクション間の制約条件Cを指定するメニューを有し、その領域にTPad貼付ることにより、それらのトランザクションが制約条件Cを持つことを定義する（図3）。Cの種類としては、2章での述べたFlexと同様に、トランザクションの成功に関する順序制約S、トランザクションの失敗に関する順序制約F、受理関数fに基づく、AND/OR等のトランザクションの同時成立制約がある。

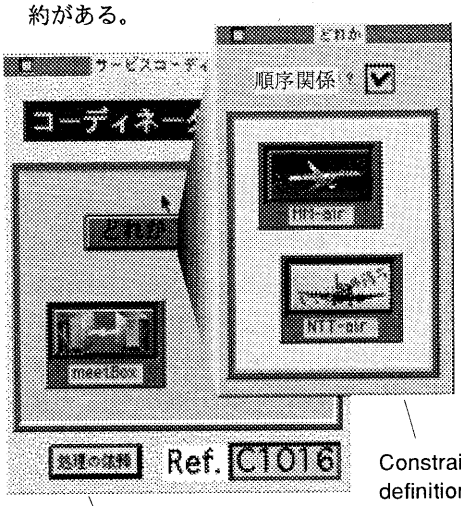

図3　Constraints definition pad

5．まとめ

本論文では、サービス間の処理の一貫性を保証するため、連携処理をトランザクションの集合とそれらの間の制約条件としてモデル化し、制約充足エンジンを用いて一貫性を保証するというアプローチをとった。また、一般利用者が容易にデータの受渡しや、制約条件を定義できるようにするため、3章で述べたように、統一的なインタフェースを提供するTPad / PPadを生成するTDFをインタフェース・ブローカによって配布するというアーキテクチャを提案するとともに、4章では、これらのpadをdrag&drop操作により貼付ることで、容易に制約条件とデータの受渡しを定義する方法を提案した。

*TxBridge*のGUIはIntelligentPadを用いてそのプロトタイプを開発したが、コーディネータは、現時点では、まだ完成していない。今後は、これらの実現と、使い勝手の評価が今後の課題である。

謝辞

IntelligentPadの使い方に関し、ご意見をいただいた日立ソフト西田宗史氏に感謝致します。

参考文献

[1] Elmagarmid, A. K. (ed.): Transaction Models for Advanced Database Applications, Morgan Kaufmann Publishers (1992).

[2] Biliris, A. el. al.: ASSET: A System for Supporting Extended Transactions, Proc. ACM SIGMOD Int'l Conf. Management of Data, pp. 44-54 (1994).

[3] Eva Kuhn, et. al.: Multidatabase Transaction and Query Processing in Logic, In book [1], pp. 297-348 (1992).

[4] Wachter, H. and Reuter, A. : The ConTract Model, In: A.K. Elmagarmid (ed.): Transaction Models for Advanced Database Applications, Morgan Kaufmann Publishers, (1992).

[5] ActiveX Controls Overview, http://www.microsoft.com/intdev/controls/controls-f.htm (1996).

[6] 長崎洋，田中譲，シンセティック・メディアシステム：IntelligentPad, コンピュータソフトウェア, Vol.11, No. 1, pp.36-48 (1994).

[7] Van Hentenryck, P., Constraint Satisfaction in Logic Programming, MIT Press (1989).